Contributors

Idea by Martin Nilsson & Michael Mutschler

Document by Martin Nilsson & Johan Sundström

Other ID3v2 contributors (Sorry if I minssed you. In order of appearance) Michael Mutschler, Christian Landgren, Steve Scherf, Ti Kan, Andreas Sigfridsson, Wataru Ishida, Kevin Bowen, Brian Stucker, Michael Ravkin, Joseph de Castelnau, Alexandre Zonine, Dave Walton, Michael Robertson, Martin Axlid, Geoffrey Elliott, Dirk Mahoney, Turadg Aleahmad, Michael Robb, Stefan Neufeind, James Webb, Guillaume Lessard, Scott Haug, Ron Goodman, Justin Rogers, Jason Molenda etc.

Domain & server by Dan O'Neill, Chief Technology Officer, MP3.com / VU Net Technologies.

This website by Martin Nilsson
ID3v2 Programming Guidelines by James Lin

Download

You can download the source code here in .ZIP format. At the moment I do not offer a compiled version.

Installation

Since you are a Java programmer, I do not need to tell you much. Simply unpack it in a directory of your choice (make sure to preserve directories), compile it, use it.

Usage

Usually, the only class you need to use is `MP3` itself. It provided a common interface to both ID3 and ID3v2 tags and to the MP3 file properties. In very special cases, e.g. if you want to read an ID3v2 tag currently not supported by `MP3` (there are two), you can directly use `ID3v2`.
Javadoc-generated class documentation is available in the `doc` subdirectory.

Note: Due to some packacking problems (one of which was size), I had to remove the `test` subdirectory for the moment.

License

The package `de.vdheide.mp3` is distributed under the GNU LIBRARY GENERAL PUBLIC LICENSE (LGPL). Please read this license at least once before using this library.

Update!!

Maintenance of this package is now the responsibility of me, Adam Russell. I hope to, in the very near future, provide an updated release. Details are forthcoming..... In the meantime for any questions please feel free to contact me.

id3lib

Latest Release Version: 3.8.2

Overview

id3lib is an open-source, cross-platform software development library for reading, writing, and manipulating ID3v1 and ID3v2 tags. It is an on-going project whose primary goals are full compliance with the ID3v2 standard, portability across several platforms, and providing a powerful and feature-rich API with a highly stable and efficient implementation.

Features

Powerful
id3lib automatically handles most of the low-level details involved with manipulating ID3v1 and ID3v2 tags in digital audio files. It provides support for several tasks associated with manipulating such tags, such as conversion between tagging formats, identifying valid tags, converting sizes, synchronisation, compression, and padding.

Standards-compliant
While many digital audio libraries and applications provide minimal support for basic ID3v1 tagging, few provide the same level of support for the up-and-coming ID3v2 standard like id3lib. The developers of id3lib work closely with the ID3v2 specification and go to great effort to ensure the library correctly handles all its nuances. By using id3lib for both your ID3v1 and ID3v2 tagging needs, you can be assured that your application produces tags that comply with the standard now and in the future.

Cross-platform
A primary goal for id3lib is cross-platform compatibility. The library is developed primarily on the GNU/Linux operating system but has been compiled and tested on other Unix-like OS's as well as Windows NT.

Multi-language
id3lib aspires to provide interfaces for multiple programming languages, and currently fully supports both C and C++. A COM wrapper (id3com) is also supplied allowing VB, VBA, VBScript and other COM-enabled languages to use the library.

Development

All development is centered around the the id3lib project page hosted by SourceForge. Please go there to find out the latest news, download the current release, join the mailing list, file a bug report, submit a patch, browse the CVS repository, or just look around.

Licensing

The id3lib library is open-source software, licensed under the GNU Library General Public License (LGPL). In short, this allows any application to link to and use the library without affecting their license, while guaranteeing that the id3lib library itself (and any modifications to it) will remain freely available in source code form. The project developers therefore greatly encourage input from everyone, be it with feature suggestions, code patches, bug reports, or anything. The best way to contribute to this effort is to subscribe to the mailing list and join in on the discussions!

Versions of ID3Lib prior to and including version 3.05a were released to the public domain. The last such release is still available for download. These versions are therefore completely free of any licence restrictions, but are no longer maintained.

Documentation

The following documentation is available for those who wish to develop software with id3lib:

- Documentation for the id3lib API is available thanks to the Doxygen documentation system.
- There is a web-readable translation of the original ID3Lib manual, written by Dirk Mahoney, the original author of ID3Lib. The API for the current library has changed somewhat, but this documentation is still mostly valid. Many thanks to Scott Moser for the web-friendly translation.
- James Lin wrote some guidelines for those programming with ID3v2 tags. It, too, is outdated, and some of the links are broken. But most of the information is still pertinent and quite valuable.

Projects

The following software projects have used, do use, or will use id3lib for their ID3v1 and ID3v2 tag processing:

- MusicMatch, creators of the MusicMatch Jukebox for Windows and a previous project coordinator for ID3Lib
- JJ MP3 Renamer is a full-featured ID3 tagger/file renamer/playlist generator.
- Idfree is a simple MP3 ID3 tag editor for Windows.
- Muzikbrowzer is an audio player application designed for the entertainment pc - a pc connected to your entertainment system! Use your remote control to browse your music collection on your tv screen! Unique music experience. Focus on your music, not the application. Runs on Microsoft Windows 2000 & XP. Supports mp3 & ogg.
- Songs-DB a Free MP3 player, jukebox and music organizer.
- FreeAmp is a cross-platform mp3 player and, as of version 2.1, is using id3lib for their ID3v1/v2 support.
- GNOMAD is a GTK+ client program for the NOMAD Jukebox. It's using id3lib for their ID3v1/v2 support.
- The id3v2 tagger is a GNU/Linux command-line editor for ID3v1/v2 tags
- Zlurp! is an all-in-one CD audio tool for Windows that rips, encodes and tags CDs.
- The javpc project's goal is to turn a PC into an audio/visual unit sutibale for plugging into a Hi-Fi and TV.
- Sonize is a collection of small but powerful tools integrated in one application that organizes and prepares your mp3 files for publishing on CDROM
- AmpBar uses a modified version of the 3.05a library in their MP3 player and ID3v2 tag editor.
- Sondra is an application that generates playlists on demand based upon rank. editor.
- UberTunz is an MP3 Music Player for Apple's new Mac OS X.
- AMIP is a the "now playing" WinAmp plugin.
- xtunes is kind of a Swiss Army Knife for enjoying digital audio on your computer.

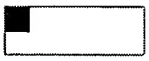
The following software projects used the original version of ID3Lib. Whether or not they still do is currently unknown.

- Easy CD-DA Extractor, a Windows CD-Ripper/Tagger/CDDB
- MacAmp, a Macintosh MP3 Player

If we have listed any of the above in error or if we have neglected any other projects that use id3lib, please let us know so that we might update the list accordingly.

Contacting the Authors

The id3lib project is collectively maintained by the id3lib Sourceforge administrators. See id3lib's Sourceforge homepage to contact them directly or join the id3lib mailing list. The original ID3Lib library was written by Dirk Mahoney.




W3C HTML 4.01

id3lib Library Documentation

3.8.0pre2

Quick Tutorial

This tutorial will quickly get you up and running with id3lib.

Downloading id3lib

First, id3lib must be a part of your development environment. The latest files can always be downloaded from the id3lib homepage.

Preparing your source code

To use the basic functionality of id3lib in your C++ code, a single #include is necessary.

```
#include <id3/tag.h>
```

There are other files that must be included to access more advanced functionality, but this will do most of the core functionality.

Creating a tag

Almost all functionality occurs via an **ID3_Tag** object. An **ID3_Tag** object basically encapsulates two things: a collection of **ID3_Frame** objects and file information. The goal is to populate an **ID3_Tag** object with **ID3_Frame** objects, and the easiest way to do this is to associate the tag with a file. This is done primarily via the **ID3_Tag** constructor, like so:

```
ID3_Tag myTag("song.mp3");
```

This constructor links, or associates, the object myTag with the file "song.mp3". In doing so, the tagging information from "song.mp3" is parsed and added to myTag. This association can also be accomplished by creating an empty tag and making an explicit call to Link().

```
ID3_Tag myTag;
myTag.Link("song.mp3");
```

The default behavior of Link() is to parse all possible tagging information and convert it into

ID3v2 frames. The tagging information parsed can be limited to a particular type (or types) of tag by passing an ID3_TagType (or combination of ID3_TagTypes). For example, to read only the ID3v1 tag, pass in the constant ID3TT_ID3V1.

```
myTag.Link("song.mp3", ID3TT_ID3V1);
```

Another example would be to read in all tags that could possibly appear at the end of the file.

```
myTag.Link("song.mp3", ID3TT_ID3V1 | ID3TT_LYRICS3V2 | ID3TT_MUSICMATCH);
```

Accessing the Tag Data

After linking with a file, the object myTag now contains some or all of the tagging information present in the file "song.mp3", represented as ID3v2 frames. How can that information be accessed? There are a variety of ways to do this. One is to iterate through all the frames in the tag.

```
// use an std::auto_ptr here to handle object cleanup automatically
ID3_Tag::Iterator* iter = myTag.CreateIterator();
ID3_Frame* myFrame = NULL;
while (NULL != (myFrame = iter->GetNext()))
{
  // do something with myFrame
}
delete iter;
```

Another way to access tagging information is by searching for specific frames using the Find() method. For example, the album frame can be found in the following manner:

```
ID3_Frame* myFrame = myTag.Find(ID3FID_ALBUM);
if (NULL != myFrame)
{
  // do something with myFrame
}
```

The Find() method can be used to search for frames with specific information. For example, the following code can be used to find the frame with the title "Nirvana".

```
ID3_Frame* myFrame = myTag.Find(ID3FID_TITLE, ID3FN_TEXT, "Nirvana"));
if (NULL != myFrame)
{
  // do something with myFrame
}
```

As indicated, the Find() method will return a NULL pointer if no such frame can be found. If more than one frame meets the search criteria, subsequent calls to Find() with the same parameters will return the other matching frames. The Find() method is guaranteed to return

all matching frames before it wraps around to return the first matching frame.

All **ID3_Frame** objects are comprised of a collection of **ID3_Field** objects. These fields can represent text, numbers, or binary data. As with frames, fields can be accessed in a variety of manners. The fields of a frame can be iterated over in much the same manner of the frames of a tag.

```
// use an std::auto_ptr here to handle object cleanup automatically
ID3_Frame::Iterator* iter = myFrame->CreateIterator();
ID3_Field* myField = NULL;
while (NULL != (myField = iter->GetNext())
{
  // do something with myField
}
delete iter;
```

If you know which field type you're looking for, you can access it directly.

```
ID3_Field* myField = myFrame->GetField(ID3FN_TEXT);
while (NULL != myField)
{
  // do something with myField
}
```

Note: The **ID3_FrameInfo** class provides information about the frame types known to id3lib.

The **ID3_Field** represents a single piece of data within an ID3v2 frame. As mentioned, an **ID3_Field** can represent three possible types of data: integers, binary data, and text strings. The type of a particular field object is immutable; it is determined at the time of its construction (almost always when a frame is constructed) and can't be changed. If in doubt, the field type can be accessed through its GetType() method.

Having an **ID3_Field** object isn't much use if you cannot access and/or alter its data. Luckily, the id3lib API provides overloaded Set and Get methods for all data types.

If the field is an integer, the following methods can be used to access the data.

```
uint32 val = myField->Get();
myField->Set(5);
(*myField) = 10;
```

All text data is accessed in a slightly different manner. The following code example best illustrates these differences.

```
// for ascii strings
char str1[1024];
const char* p1 = "My String";
const char* p2 = "My Other String";

myField->Set(p1);
(*myField) = p2;  // equivalent to Set

myField->Get(str1, 1024); // copies up to 1024 bytes of the field data into str1
p1 = myField->GetRawText(); // returns a pointer to the internal string
```

Binary data is similar to text data, except that its base type is a pointer to an unsigned, rather than a signed, char.

```
// for binary strings
uchar data[1024];
const uchar *p1 = getBinaryData(); // not an id3lib function
size_t size = getBinarySize();     // not an id3lib function

myField->Set(p1, size);

myField->Get(data, 1024); // copies up to 1024 bytes of the field data into str1
p1 = myField->GetRawBinary(); // returns a pointer to the internal string
```

Updating the Tag

When you're ready to save your changes back to the file, a single call to Update() is sufficient.

```
tag.Update();
```

ID3v2 format Search

Search again with these terms:
apa format | mla format | resume format | formatting disk

Search Tips

Sponsored Links About This

1-10 of 10601 < Previous 10 | Next 10 >

- **Dr.TAG - MP3 Tagger**
  Tag, rename, sort, organize and manage your whole MP3 collection!
  www.drtag.de

Matching Results About This

1. **Replay Gain - File format ID3v2 mp3 information**
   Replay Gain - A Proposed Standard. Replay Gain ID3v2 File Format. Existing format. The ID3v2 format is explain at www.id3.org. ...
   http://replaygain.hydrogenaudio.org/f...

2. **MP3-Tag-Editor - FAQ**
   ... nicht mehr. Titelinformationen im ID3v2-Format k□nnen nahezu unbegrenzt Felder mit unterschiedlichsten Inhalten enthalten. Eine MP3 ...
   http://www.mp3-tag-editor.de/faq.php

3. **Mp3/Tag Studio**
   ... So, this was exacly what a couple of guys did, and the results were the ID3v2 format. The ID3v2 tag is fundamentally different from ID3v1 tags. ...
   http://www.magnusbrading.com/mp3ts/id...

4. **Project Menu**
   ... Audacity allows you to store ID3 tags in either the ID3v1 or the ID3v2 format. In general, you should use the ID3v2 format, because ...
   http://audacity.sourceforge.net/docs1...

5. **MP3::Tag::ID3v2-Data - get_frame data format and supporte...**
   ... This document describes how to use the results of the get_frame function of MP3::Tag::ID3v2, thus the data format of frames retrieved with MP3::Tag::ID3v2 ...
   http://tagged.sourceforge.net/id3v2-D...

6. **MP3 Support in Windows XP**
   ... Player and get full support for all of Windows XP□s new digital media features including automatic album art, media information in ID3v2 format and much more ...
   http://www.microsoft.com/windows/wind...

7. **Windows Media and MP3s**
   ... Player and get full support for the same digital media features enjoyed by WMA users including automatic album art, media information in ID3v2 format and much ...
   http://www.microsoft.com/windows/wind...

8. **PZ TagEditor ID3v2**
   PZ. TagEditor. Main, Since versiov 4.0 PZ TagEditor support ID3v2 format. Its main changes from ID3v1: - big sizes of fields (groups ...
   http://www.pztageditor.com/id3v2_e.htm

9. **ActiveVB - Tipp 0408: ID3v2-Tag aus MP3s auslesen**
   ... Eintr□, sowie nur sechs unterschiedliche Eintr□ (Titel, K□nstler, Album, Jahr, Kommentar und Genre - wurde das erweiterte ID3v2-Format eingef□hrt, was ...
   http://www.activevb.de/tipps/vb6tipps...

10. **Tag format specs - MP3 Manager**
    ... ETT 00250 Extended Track Title. IMG 99999 Link to an image files (BMP, JPG, GIF format). ... ID3v2 Lyrics3 tags. About Lyrics3 Tag version 2.00 and ID3v2. ...
    http://www.volweb.cz/str/tags.htm

1-10 of 10601 < Previous 10 | Next 10 >

Search Again

Search

Search again with these terms:
apa format | mla format | resume format | formatting disk

Search Tips

Search our affiliates: Lycos | Ask Jeeves | Overture | Looksmart

Replay Gain - A Proposed Standard

Replay Gain ID3v2 File Format

Existing format

The ID3v2 format is explain at www.id3.org. The most useful document is the ID3v2 v2.3.0 standard*. Note that all ID3v2 tag frames are written in Big Endian Byte order.

* Whilst this document has been superseded by v2.4.0 (see ID3v2 Developer Information) the earlier document is complete (rather than an update), and in indexed HTML form. As such, it represents a better technical introduction to ID3v2.

The ID3v2 standard defines a "tag" which is situated *before* the data in an mp3 file. The original ID3 (v1) tags resided at the end of the file, and contained a few fields of information. The ID3v2 tags can contain virtually limitless amounts of information, and new "frames" within the tags may be defined.

Replay Gain Adjustment chunk

In the language of the ID3v2 standard document, a new frame is suggested, thus:

```
<Header for 'Replay Gain Adjustment', ID: "RGAD">
Peak Amplitude                          $xx $xx $xx $xx
Radio Replay Gain Adjustment            $xx $xx
Audiophile Replay Gain Adjustment       $xx $xx

Header consists of:
Frame ID                $52 $47 $41 $44 = "RGAD"
Size                    $00 $00 $00 $08
Flags                   $40 $00         (%01000000 %00000000)
```

In the RGAD frame, the flags state that the frame should be preserved if the ID3v2 tag is altered, but discarded if the audio data is altered.

Does it work?

I haven't tried this myself. Some example code for parsing ID3v2 tags is available here, which could easily be extended to write the above tag. Within-range Replay Gain data **can** contain the synchronisation code ($FF), so unsynchronisation may be necessary for this frame.

Suggestions and Further Work

If you can see any problem with this proposal, please let me know. No software implements this (as yet).

◂ [File Format] [CONTENTS] [Player Requirements] ▸

PZTagEditor

Main
ID3v2
Screenshot
F.A.Q.
History
Download
Register
Guestbook

Since versiov 4.0 PZ TagEditor support ID3v2 format.
Its main changes from ID3v1:
- big sizes of fields (groups of ten Mb) allows save all that necessary, not taking care of bounds of fields;
- except six floors, supported by versions 1.x, there is several additional "standard" fields (Composer, Equalizer Adjustment, Volume and etc.) and it is provided possibility to add their own fields;
- possibility to save jpg- or other pictures to ID3-fields;
- ...

More detailed information on ID3v2 is kept on official site of ID3: http://www.id3.org

Set of supported ID3v2-fields, given by current version of program, while it is limited standard six fields (Artist, Title, Album, Year, Genre, Comments). Support of other ID3v2-fields will be included in following versions of PZ TagEditor on measure of origin to need.

ID3v2

The audience is informed

Introduction
A short explanation of ID3v2 and related standards

In depth information
Technical specifications and binaries.
News: ID3v2.4.0 officially released

Implementations
Existing and future programs and sources.

Contributors

Mailinglist

Tagging introduction

Low tech

The short history of tagging - A quick background to what MP3 and ID3 are.
ID3v2 made easy - A non-technical introduction to ID3v2.

Mid tech

ID3 made easy - A short description of ID3 (v1 and v1.1).
Lyrics3 made easy - A quick look at a tagging format for lyrics.
Lyrics3 v2.00 - A closer look to the latest Lyrics3 standard.
The private life of MP3 frames - How does the internal of an MP3 file look like.

The short history of tagging

The past

Once upon a time, there were some giant companies that, with the failure of the 4-channel battle fresh in mind, formed an expert group with the mission to invent tomorrow's technology in sound compression. Fortunately, they did. The format, named MPEG Layer 3 or for short MP3, took advantage of the fact that our ears are not nearly as good as we generally believe them to be, and thus omitting frequencies that we wouldn't hear anyway. They also made the format suitable for streaming by letting the sound be represented in small, individually compressed blocks of audio data. Each block had a header containing some information relevant to the decoding process. As they ended up with a few bits to much, they used them for some additional information such as a 'copyright' bit and a 'private' bit.

Since the format had such an outstanding compression and still very good sound quality, it was soon adapted as the de facto standard for digital music. The lack of possibilities to include textual information in the files was however disturbingly present. Suddenly, someone (Eric Kemp alias NamkraD, I've been told) had a vision of a fix-sized 128-byte tag that would reside at the end of the audio file. It would include title, artist, album, year, genre and a comment field. Someone, possibly the very same someone, implemented this and everyone was happy. Soon afterwards, Michael Mutschler, the author of MP3ext, extended this tag, called ID3, to also include which track on the CD the music originated from. He used the last two bytes of the comment field for this and named his variant ID3 v1.1. (more information about ID3 and ID3v1.1 can be found **here**)

The present

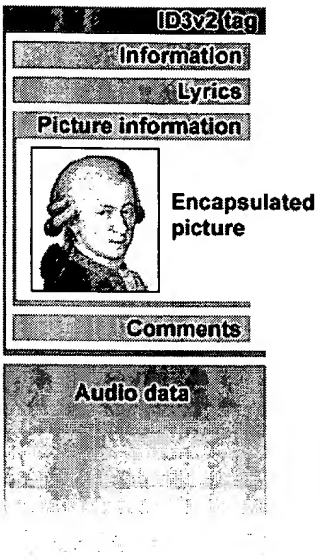
The ID3 v1.1 tag still had some obvious limitations and drawbacks, though. It supported only a few fields of information, and those were limited to 30 characters, making it impossible to correctly describe "The Hitchhiker's Guide to the Galaxy from BBC Radio" as well as "P.I. Tchaikovsky's Nutcracker Suite Op. 71 a, Ouverture miniature danses caractéristiques by The New Philharmonic Orchestra, London, conducted by Laurence Siegel". Since the position of the ID3 v1.1 tag is at the end of the audio file it will also be the last thing to arrive when the file is being streamed. The fix size of 128 bytes also makes it impossible to extend further. That's why I (Martin Nilsson) and several along with me thought that a new ID3 tag would be appropriate.

The new ID3 tag is named ID3v2 and is currently in a state of 'informal standard'. That is, we decided, since there were less and less improvements and additions made, to proclaim the draft as a standard (an informal one since no standardization body has approved this decision). You can find the informal standard **here**. ID3v2 is often followed by its revision number, i.e. the current informal standard is ID3v2.4.0.

ID3v2 made easy

What is ID3v2?

ID3v2 is a new tagging system that lets you put enriching and relevant information about your audio files within them. In more down to earth terms, ID3v2 is a chunk of data prepended to the binary audio data. Each ID3v2 tag holds one or more smaller chunks of information, called frames. These frames can contain any kind of information and data you could think of such as title, album, performer, website, lyrics, equalizer presets, pictures etc. The block scheme to the right is an example of how the layout of a typical ID3v2 tagged audio file may look like.



Example of the internal layout of an ID3v2 tagged file.

One of the design goals were that the ID3v2 should be very flexible and expandable. It is very easy to add new functions to the ID3v2 tag, because, just like in HTML, all parsers will ignore any information they don't recognize. Since each frame can be 16MB and the entire tag can be 256MB you'll probably never again be in the same situation as when you tried to write a useful comment in the old ID3 being limited to 30 characters.

Speaking of characters, the ID3v2 supports Unicode so even if you use the Bopomofo character set you'll be able to write in your native language. You can also include in which language you're writing so that one file might contain e.g. the same lyrics but in different languages.

Even though the tag supports a lot of byte consuming capabilities like inline pictures and even the possibility to include any other file, ID3v2 still tries to use the bytes as efficient as possibly. If you convert an ID3v1 tag to an ID3v2 tag it is even likely that the new tag will be smaller. If you convert an ID3v1 tag where all fields are full (that is, all 30 characters are used in every field) to an ID3v2 tag it will be 56 bytes bigger. This is the worst case scenario for ID3v1 to ID3v2 conversion.

Since it's so easy to implement new functionality into ID3v2, one can hope that we'll see a lot of creative uses for ID3v2 in the future. E.g. there is a built-in system for rating the music and counting how often you listen to a file, just to mention some brainstorm results that are included. This feature can be used to build playlists that play your favourite songs more often than others.

Some main features

- The ID3v2 tag is a container format, just like IFF or PNG files, allowing new frames (chunks) as evolution proceeds.
- Residing in the beginning of the audio file makes it suitable for streaming.
- Has an 'unsynchronization scheme' to prevent ID3v2-incompatible players to attempt to play the tag.
- Maximum tag size is 256 megabytes and maximum frame size is 16 megabytes.
- Byte conservative and with the capability to compress data it keeps the files small.
- The tag supports Unicode.
- Isn't entirely focused on musical audio, but also other types of audio.
- Has several new text fields such as composer, conductor, media type, BPM, copyright message, etc. and the possibility to design your own as you see fit.
- Can contain lyrics as well as music-synced lyrics (karaoke) in almost any language.
- Is able to contain volume, balance, equalizer and reverb settings.
- Could be linked to CD-databases such as CDDB.
- Is able to contain images and just about any file you want to include.
- Supports enciphered information, linked information and weblinks.
- and more... (a complete list of all frames and their functions can be found **here**)

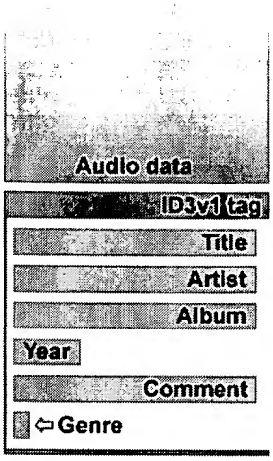
ID3 made easy

What is ID3 (v1)?

The audio format MPEG layer I, layer II and layer III (MP3) has no native way of saving information about the contents, except for some simple yes/no parameters like "private", "copyrighted" and "original home" (meaning this is the original file and not a copy). A solution to this problem was introduced with the program "Studio3" by Eric Kemp alias NamkraD in 1996. By adding a small chunk of extra data in the end of the file one could get the MP3 file to carry information about the audio and not just the audio itself.

The placement of the tag, as the data was called, was probably chosen as there were little chance that it should disturb decoders. In order to make it easy to detect a fixed size of 128 bytes was chosen. The tag has the following layout (as hinted by the scheme to the right):

Song title	30 characters
Artist	30 characters
Album	30 characters
Year	4 characters
Comment	30 characters
Genre	1 byte



Internal layout of an ID3v1 tagged file.

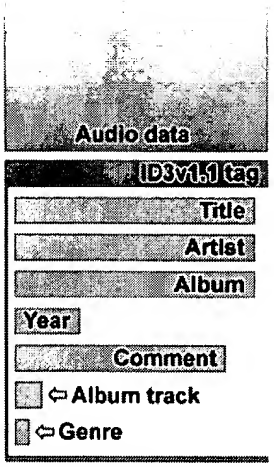
If you one sum the the size of all these fields we see that 30+30+30+4+30+1 equals 125 bytes and not 128 bytes. The missing three bytes can be found at the very end of the tag, before the song title. These three bytes are always "TAG" and is the identification that this is indeed a ID3 tag. The easiest way to find a ID3v1/1.1 tag is to look for the word "TAG" 128 bytes from the end of a file.

As all artists doesn't have a 30 character name it is said that if there is some bytes left after the information is entered in the field, those bytes should be fille with the binary value 0. You might also think that you cannot write that much in the genre field, being one byte big, but it is more clever than that. The byte value you enter in the genre field corresponds to a value in a predefined list. The list that Eric Kemp created had 80 entries, ranging from 0 to 79.

What is ID3v1.1?

ID3v1 may well be easy to implement for programmers, but it sure is frustrating for those with their own, creative ideas. Since the ID3v1 tag had a fixed size and no space marked "Reserved for future use", there isn't really room for that much improvement, if you want to maintain compatibility with existing software.

One who found a way out was Michael Mutschler who made a quite clever improvement on ID3v1. Since all non-filled fields must be padded with zeroed bytes its a good assumption that all ID3v1 readers will stop reading the field when they encounter a zeroed byte. If the second last byte of a field is zeroed and the last one isn't we have an extra byte to fill with information. As the comments field is to short to write anything useful in the ID3v1.1 standard declares that this field should be 28 characters, that the next byte always should be zero and that the last byte before the genre byte should contain which track on the CD this music comes from.



Internal layout of an ID3v1.1 tagged file.

Lyrics3 made easy

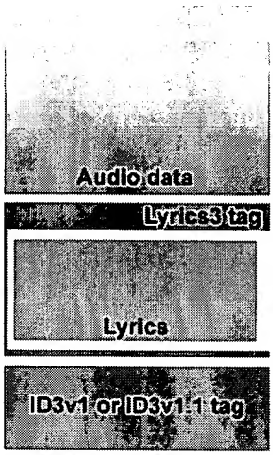
What is Lyrics3?

When Winamp introduced its plugin capabilities Kuo (Djohan) Shiang-shiang's Lyrics Displayer was one of the first plugins, and probably the first program that made a connection between MP3 audio and lyrics. The lyrics displayed by Kuo's Lyrics Displayer were stored in separate text files from which the program got it.

Petr Strnad saw the problems in this so he decided to make a lyrics tag, enabling the text to reside inside the audio file. This is done by creating a chunk of data which begins with "LYRICSBEGIN", ends with "LYRICSEND" and has the lyrics between these keywords. This data block is then saved in the audio file between the audio and the ID3 tag. If no ID3 tag is present one must be attached.

The following simple rules applies to the lyrics inserted between the keywords:

- The keywords "LYRICSBEGIN" and "LYRICSEND" must not be present in the lyrics.
- The text is encoded with ISO-8859-1 character set
- A byte in the text must not have the binary value 255.
- The maximum length of the lyrics is 5100 bytes.
- Newlines are made with CR+LF sequence.



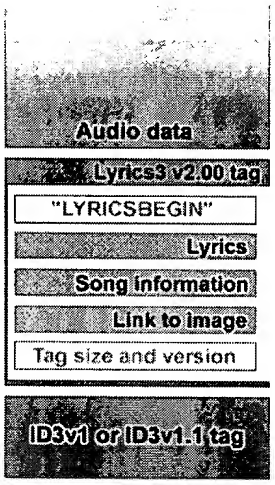
Internal layout of an Lyrics3 tagged file.

Lyrics3 Tag v2.00

Based on official specification dated Jun 5, 1998

What is Lyrics3 v2.00?

The Lyrics3 v2.00 tag is more complicated than the previous Lyrics3 tag but has a lot more capabilities. Just like the old Lyrics3 tag it resides between the audio and the ID3 tag, which must be present. The tag uses only text for everything from content to size descriptors, which are represented as numerical strings. This makes it easier to implement a Lyrics3 v2.00 reader. At least if BASIC is your language of choice.



The Lyrics3 block, after the MP3 audio and before the ID3 tag, begins with the word "LYRICSBEGIN" after which a number of field records follows. The Lyrics3 block ends with a six character size descriptor and the string "LYRICS200". The size value includes the "LYRICSBEGIN" and "LYRICS200" strings.

Lyrics2 v2.00 uses somthing called fields to represent information. The data in a field can consist of ASCII characters in the range 01 to 254 according to the standard. As the ASCII character map is only defined from 00 to 128 ISO-8859-1 might be assumed. Numerical fields are 5 or 6 characters long, depending on location, and are padded with zeroes.

Only the size of the tag sets the limit for how many fields may be present. All fields uses a simple structure that includes a three character field ID, six characters describing the size of the information and the actual information. This makes it possible to read unknown fields and write them back when saving the tag. There are no required fields in the tag, but at least one field must exist. Fields can appear in any order in the tag, except the indication field which must be the first field if used. Fields that include more then one line uses [CR][LF] delimiters between lines.

Defined fields

The following list is a list of currently defined field IDs. More fields might be added if needed on newer versions of the Lyrics3 v2.00 specifications. Unknown fields should be ignored.

ID	Max size	Description
IND	00002	Indications field. This is always two characters big in v2.00, but might be bigger in a future standard. The first byte indicates wether or not a lyrics field is present. "1" for present and "0" for otherwise. The second character indicates if there is a timestamp in the lyrics. Again "1" for yes and "0" for no.
LYR	99999	Lyrics multi line text. Timestamps can be used anywhere in the text in any order. Timestamp format is [mm:ss] (no spaces allowed in the timestamps).
INF	99999	Additional information multi line text.
AUT	00250	Lyrics/Music Author name.
EAL	00250	Extended Album name.
EAR	00250	Extended Artist name.
ETT	00250	Extended Track Title.
IMG	99999	Link to an image files (BMP or JPG format). Image lines include filename, description and timestamp separated by delimiter - two ASCII chars 124 ("||"). Description and timestamp are optional, but if timestamp is used, and there is no description, two delimiters ("||||") should be used between the filename and the timestamp. Multiple images are allowed by using a [CR][LF] delimiter between each image line. No [CR][LF] is needed after the last image line. Number of images is not limited (except by the field size). **Filename** can be in one of these formats: • Filename only - when the image is located in the same path as the MP3 file (preferred, since if you move the mp3 file this will still be correct) • Relative Path + Filename - when the image is located in a subdirectory below the MP3 file (i.e. images\cover.jpg) • Full path + Filename - when the image is located in a totally different path or drive. This will not work if the image is moved or drive letters has changed, and so should be avoided if possible (i.e. c:\images\artist.jpg) **Description** can be up to 250 chars long. **Timestamp** must be formatted like the lyrics timestamp which is "[mm:ss]". If an image has a timestamp, then the visible image will automatically switch to that image on the timestamp play time, just the same as the selected lyrics line is switched based on timestamps.

The extended Album, Artist and Track are an extension to the fields in the ID3v1 tag - which are limited to 30 chars. If these extended fields exist, make sure their first 30 chars are exactly the same as the ones in the ID3v1 tag. If they are the same, display the extended field. If not, display the one from the ID tag. These 'mismatched' extended fields, should be removed when saving the lyrics tag.

When saving the extended fields, make sure to copy the first 30 chars of each field to the ID3 tag matching fields. It is recommended NOT to save extended fields at all, if they are not larger then 30 chars.

Example

The following includes all the information from after the mp3 data until the ID tag. Please note that unless a [CR][LF] is at the end of the line, no [CR] or [LF] should be added between lines.

```
LYRICSBEGIN
IND00002
11
EAL00041
Album name that is larger then 30 chars !
EAR00050
Artist name or band that is larger then 30 chars !
ETT00042
Track name which is larger then 30 chars !
INF00090
This track was actually recorded in several places around the world[CR][LF]
and mixed at the US[CR][LF]
AUT00048
The lyrics were written by someone.. is it you ?
IMG00086
album_cover.jpg||Album cover|||[00:10][CR][LF]
jumping.jpg||He jumps at the audience!||[01:00]
LYR00630
[00:02]Let's talk about time[CR][LF]
[00:02]tickin' away every day[CR][LF]
[00:05]so wake on up before it's gone away[CR][LF]
[00:10]catch the 411 and stay up like the sun[CR][LF]
[00:20]remind yourself what's done and done[CR][LF]
[00:32]so let yesterday stay with the bygones[CR][LF]
[00:40]keep your body and soul and your mind on[CR][LF]
[00:55]the right track infact you gotta stay on[CR][LF]
[01:20]the real black[CR][LF]
[CR][LF]
Chorus:[CR][LF]
[01:25][05:45]Time is tickin' away[CR][LF]
[01:42][05:55]you've gotta - live your life -[CR][LF]
[02:11][06:24]day by day[CR][LF]
[02:26][06:35]happy or sad, good or bad[CR][LF]
[02:31][06:42]life is too short[CR][LF]
[02:58][07:13]you've gotta - keep your head -[CR][LF]
[03:01][07:19](Repeat)[CR][LF]
001064LYRICS200
```

The indications field size is two bytes. They are two '1', which mean that LYR field is to be found in the tag, and timestamps are used in it.
The extended Album text size is 41 chars
The extended Artist name size is 50 chars
The extended Track name size is 42 chars
The Additional information size is 90 chars
The Lyrics Author size is 48 chars
The Image file link size is 86 chars
The Lyrics text size is 630 chars
All the fields together (with the 'LYRICSBEGIN') is 001064 chars.

The ID3v1 tag that comes after this example should have :
Album field: 'Album name that is larger then'
Artist field: 'Artist name or band that is la'
Title field: 'Track name which is larger the'

Contact information

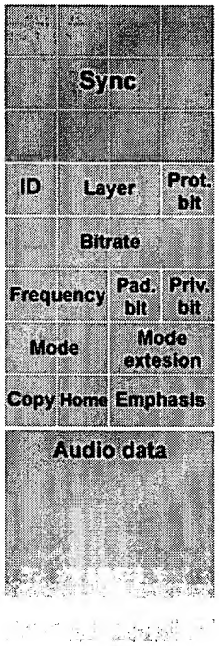
The Lyrics3 v2.00 tag specification was written by Strnadp and Alon Gingold.

The private life of MP3 frames

How is MP3 built?

Most people with a little knowledge in MP3 files know that the sound is divided into smaller parts and compressed with a psycoacoustic model. This smaller pieces of the audio is then put into something called 'frames', which is a little datablock with a header. I'll focus on that header in this text.

The header is 4 bytes, 32 bits, big and begins with something called sync. This sync is, at least according to the MPEG standard, 12 set bits in a row. Some add-on standards made later uses 11 set bits and one cleared bit. The sync is directly followed by a ID bit, indicating if the file is a MPEG-1 och MPEG-2 file. 0=MPEG-2 and 1=MPEG-1



Frame header.

The layer is defined with the two layers bits. They are oddly defined as

0 0	Not defined
0 1	Layer III
1 0	Layer II
1 1	Layer I

With this information and the information in the bitrate field we can determine the bitrate of the audio (in kbit/s) according to this table.

Bitrate value	MPEG-1, layer I	MPEG-1, layer II	MPEG-1, layer III	MPEG-2, layer I	MPEG-2, layer II	MPEG-2, layer III
0 0 0 0						
0 0 0 1	32	32	32	32	32	8
0 0 1 0	64	48	40	64	48	16
0 0 1 1	96	56	48	96	56	24
0 1 0 0	128	64	56	128	64	32
0 1 0 1	160	80	64	160	80	64
0 1 1 0	192	96	80	192	96	80
0 1 1 1	224	112	96	224	112	56
1 0 0 0	256	128	112	256	128	64
1 0 0 1	288	160	128	288	160	128
1 0 1 0	320	192	160	320	192	160
1 0 1 1	352	224	192	352	224	112
1 1 0 0	384	256	224	384	256	128
1 1 0 1	416	320	256	416	320	256
1 1 1 0	448	384	320	448	384	320
1 1 1 1						

The sample rate is described in the frequency field. These values is dependent of which MPEG standard is used according to the following table.

Frequency value	MPEG-1	MPEG-2
0 0	44100 Hz	22050 Hz
0 1	48000 Hz	24000 Hz
1 0	32000 Hz	16000 Hz
1 1		

Three bits is not needed in the decoding process at all. These are the copyright bit, original home bit and the private bit. The copyright has the same meaning as the copyright bit on CDs and DAT tapes, i.e. telling that it is illegal to copy the contents if the bit is set. The original home bit indicates, if set, that the frame is located on its original media. No one seems to know what the privat bit is good for.

If the protection bit is NOT set then the frame header is followed by a 16 bit checksum, inserted before the audio data. If the padding bit is set then the frame is padded with an extra byte. Knowing this the size of the complete frame can be calculated with the following formula

FrameSize = 144 * BitRate / SampleRate
when the padding bit is cleared and

FrameSize = (144 * BitRate / SampleRate) + 1
when the padding bit is set.

The frameSize is of course an integer. If for an example BitRate=128000, SampleRate=44100 and the padding bit is cleared, then the FrameSize = 144 * 128000 / 44100 = 417

The mode field is used to tell which sort of stereo/mono encoding that has been used. The purpose of the mode extension field is different for different layers, but I really don't know exactly what it's for.

Mode value	mode
0 0	Stereo
0 1	Joint stereo
1 0	Dual channel
1 1	Mono

The last field is the emphasis field. It is used to sort of 're-equalize' the sound after a Dolby-like noise supression. This is not very used and will probably never be. The following noise supression model is used

Emphasis value	Emphasis method
0 0	none
0 1	50/15ms
1 0	
1 1	CCITT j.17

Implementations

ID3 C/C++ Library

An ID3 C/C++ library, named ID3Lib, is currently in development. The sourcecode is coordinated by Scott Haug and was initially written by Dirk Mahoney and Andreas Sigfridsson.

The ID3Lib page.

Java class MP3

Jens Vonderheide has written a java class that handles ID3v1, ID3v2 and MP3 properties. Currently hosted and developed by Adam Russell.

ID3 java

Perl classes

Both Chris Nandor and Matt DiMeo has some promising perl classes on cpan.

MP3::Info
MPEG::ID3v2Tag

ActiveX components

ShazamMP3
EzyID3

Borland Delphi library

James Webb has made a Delphi libraray with which you can read and write ID3v2.3.0 and ID2.4.0.

ID3v2 Delphi Library

Other implementations

A few programs that supports ID3v2. If you know of any ID3v2 supporting software that is not on the list, please contact us. NOTE: These programs have not been checked for ID3v2 complience by id3.org.

AmigaAMP
Argon
AudioCatalog
Audiograbber
Audion 2
CD-Copy
CDDB-MP3-Tagger v2.0
Dr.TAG
Easy CD-DA Extractor
FreeAmp
Helium
HTagEditor
ID3-TagIT
The ID3v2 Package
ID3v2 Tag Edit
iTunes
MP3 Boss
MP3 Collector
MP3 Explorer
MP3 Rage
MP3 Tag Studio
MP3Browser
MP3Ext
MPTagger
Music Library
Music Match
QuickTime 5
SingedIT
SoundJam
Tag&Rename
Winamp
ZLURP

PZTagEditor

Main
ID3v2
Screenshot
F.A.Q.
History
Download
Register
Guestbook

4.40 - 01.11.2002

Added:

-Reading/writing of extended fields: composer, copyrights, URL, publisher...;
- Lyrics reading/writing;
- Option "Never add ID3v2" on saving;
- ID3-fields case formatting;
- ID3-fields transliterating (for non english languages);
- "Don't rewrite already existent fields" option on Filename->ID3;
- Tranliteration for any non-english languages support;
- No changes of file modifing date when you save ID3;
- ...

Changed:

- ...

Fixed:

- VBR-files invalid duration when ID3v2 has been edited;
- ...

4.30 - 20.02.2002

Added:

- subfolders scanning;
- "Track" field reading/writing;
- playlists template adjusting;
- strings formatting;
- undo current changes;
- Encoder reading;
- menu "Select All" and "Registration";
- folder creating from Copy/Move dialog possibility;
- one copy of program from one folder;
- reading of settings of previous version of program;
- ...

Changed:

- Track number in Advanced Renaming and creating ID3 from filenames dialogs;
- Folder, Track, Encoder columns in filelist;
- unchanging of file's date;
- ...

Fixed:

- playing of files from root folder;
- start program from folders/files with long names;
- comments reading/writing error;
- playing of files with external players;
- ...

4.20 - 27.10.2001

Added:

- ID3-fields from filenames creating
- export data to Microsoft Excel
- multilanguage interface
- M3U-playlists creating
- working with network folders
- copy/move files
- several file extensions in MPEG-mask
- ID3 saveing during playback
- "If Not Exist ID3v2 - Add" option ("Always", "If It's Need")
- break ReadOnly attribute on ID3 saving
- replace incorrect symbols to "_" when rename files

Changed:

- set cursor to selected file from Windows Explorer popup menu
- new advanced file renaming

- display playtime with MIN:SEC format
- append records to HTM-playlist possibility
- warning dialog on filelist changing without saving

Fixed:

- columns adjusting error with autowidth option
- root folder error
- MP3 header error on ID3v2 saving
- ID3v1 fields with length more 30 characters displaying
- sort parameters when folder is changed
- ...

4.1.1 - 24.03.2001

Fixed:

- working speed is optimized
- displying error in "Small Icons" and "List" mode
- focus positioning error
- ...

4.1 - 25.02.2001

Changed:

- only one button for all ID3-fields saving
- work with Winamp is optimized

Fixed:

- group of files deleting
- play next file timer
- files open fith external players
- scroll file list to selected file
- refreshing error
- ID3 version sorting
- error on F2 file renaming
- filelist error with empty folder
- ...

4.0 - 25.01.2001

Added:

- ID3v2 support
- copy ID3: from v1 to v2, from v2 to v1
- ID3 version column in filelist

Changed:

- remove ID3: v1 or v2

3.3 - 11.01.2001

Added:

- columns adjustment
- step to next file after saving any ID3-field
- cyrillic <-> latin translit on ID3 and filenames
- 10 sec. forward with internal player
- step to Prev/Next file with hotkeys
- help system

Fixed:

- ...

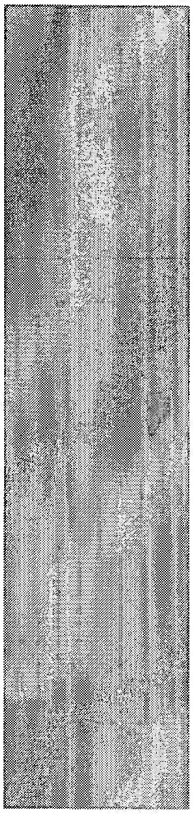
3.2 - 01.11.2000

Added:

- columns autosize
- VBR recognition
- sort filelist by any column
- changes highlighting

Fixed:

- ...



3.0 - 21.08.2000
Added:
- all ID3-fields in filelist viewing
- advanced renaming with ID3-fields
- PLS- and HTM-playlists creating

Fixed:
- error message on program start in Windows NT/2000
- ID3 reading from files on CD
- ...

2.0 - 25.04.2000
Added:
- displays MPEG Layer, Bitrate, Playtime
- internal player
- ...

Changed:
- windows Explorer-like interface of main window
- ...

1.0 - 02.08.1999
Overview:
- Viewing and editing ID3v1-fields of MP3 files.
- Step to next file on "Title" field saving. Other ID3-fiels saves with dialog: "Save all or selected only files?".
- Deleting ID3v1.
- Play file with associated application.
- Rename files with template "Track##.mp3".

Documentation

The following documentation is available for those who wish to develop software with id3lib:

- Documentation for the id3lib API is available thanks to the Doxygen documentation system.
- There is a web-readable translation of the original ID3Lib manual, written by Dirk Mahoney, the original author of ID3Lib. The API for the current library has changed somewhat, but this documentation is still mostly valid. Many thanks to Scott Moser for the web-friendly translation.
- James Lin wrote some guidelines for those programming with ID3v2 tags. It, too, is outdated, and some of the links are broken. But most of the information is still pertinent and quite valuable.

Projects

The following software projects have used, do use, or will use id3lib for their ID3v1 and ID3v2 tag processing:

- MusicMatch, creators of the MusicMatch Jukebox for Windows and a previous project coordinator for ID3Lib
- JJ MP3 Renamer is a full-featured ID3 tagger/file renamer/playlist generator.
- Idfree is a simple MP3 ID3 tag editor for Windows.
- Muzikbrowzer is an audio player application designed for the entertainment pc - a pc connected to your entertainment system! Use your remote control to browse your music collection on your tv screen! Unique music experience. Focus on your music, not the application. Runs on Microsoft Windows 2000 & XP. Supports mp3 & ogg.
- Songs-DB a Free MP3 player, jukebox and music organizer.
- FreeAmp is a cross-platform mp3 player and, as of version 2.1, is using id3lib for their ID3v1/v2 support.
- GNOMAD is a GTK+ client program for the NOMAD Jukebox. It's using id3lib for their ID3v1/v2 support.
- The id3v2 tagger is a GNU/Linux command-line editor for ID3v1/v2 tags
- Zlurp! is an all-in-one CD audio tool for Windows that rips, encodes and tags CDs.
- The javpc project's goal is to turn a PC into an audio/visual unit sutibale for plugging into a Hi-Fi and TV.
- Sonize is a collection of small but powerful tools integrated in one application that organizes and prepares your mp3 files for publishing on CDROM
- AmpBar uses a modified version of the 3.05a library in their MP3 player and ID3v2 tag editor.
- Sondra is an application that generates playlists on demand based upon rank. editor.
- UberTunz is an MP3 Music Player for Apple's new Mac OS X.
- AMIP is a the "now playing" WinAmp plugin.
- xtunes is kind of a Swiss Army Knife for enjoying digital audio on your computer.

The following software projects used the original version of ID3Lib. Whether or not they still do is currently unknown.

- Easy CD-DA Extractor, a Windows CD-Ripper/Tagger/CDDB
- MacAmp, a Macintosh MP3 Player

If we have listed any of the above in error or if we have neglected any other projects that use id3lib, please let us know so that we might update the list accordingly.

Contacting the Authors

The id3lib project is collectively maintained by the id3lib Sourceforge administrators. See id3lib's Sourceforge homepage to contact them directly or join the id3lib mailing list. The original ID3Lib library was written by Dirk Mahoney.


SOURCEFORGE
.net


W3C HTML 4.01